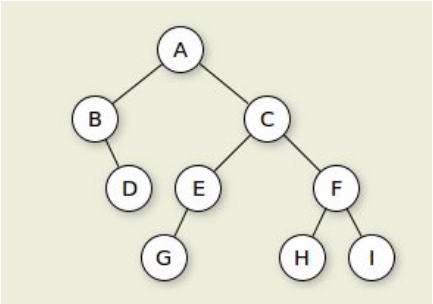
Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos I
Segundo Cuatrimestre de 2016

Práctica No. 6 (Árboles)

Ej. 0. Dado el siguiente árbol binario:



- Marcar el nodo raíz
- Cuántos y Cuáles son nodos hoja?
- Cuáles nodos son ancestros del nodo G?
- Cuál es la altura del árbol?
- Cuántos y Cuáles son los nodos del nivel 3?
- Cuál es la profundidad del nodo G?

Ej. 1. Implemente la interface Tree, vista en clases.
Ej. 2. Tenemos un árbol binario t, su recorrido preorden es GEAIBMCLDFKJH y en inorden IABEGLDCFMKHJ, dibujar el árbol, y dar su recorrido posorden.
Ej. 3. Agregue los métodos preorder, inorder, postorder en su clase Tree.
Ej. 4. Implemente la clase ABB, de árboles binarios de búsqueda con los métodos insert, delete, search y RepOk.
Para cada método cálcule su tiempo de ejecución en el peor caso. (En los comentarios de la clase debe incluir un comentario para decir que orden es su algoritmo).
Ej. 5. implemente la clase Ntree en Java, la clase NTree implementa los árboles n-arios. Defina al menos dos formas de recorrer sus árboles.
Ej. 6. Usando su clase ABB, implemente el algoritmo TreeSort visto en clases. Compare este algoritmo con el resto de la clase sorting.
Ej. 7. Implemente la clase Heap con las operaciones insertar, remover, esVacio y repOk. Para cada método cálcule su tiempo de ejecución en el peor caso.
Ej. 8.[Opcional] Utilizando su clase Heap implemente el algoritmo HeapSort (agreguelo en la clase sorting de la práctica anterior). La idea del algoritmo es construir un heap con los elementos del arreglo a ordenar, para luego utilizar el remover para construir el arreglo resultante. Compare este algoritmo con el resto de la clase Sorting.
Ej. 9. Supongamos que poseemos una implementación de AVL's, empezando desde el árbol vacío, ilustrar como va quedando el AVL cuando se ejecutan las siguientes operaciones:

- t.insert(10)
- t.insert(100)

- t.insert(30)
- t.insert(80)
- t.insert(50)
- t.delete(10)
- t.insert(60)
- t.insert(70)
- t.insert(40)
- t.delete(80)
- t.insert(90)
- t.insert(20)
- t.delete(30)
- t.delete(70)

Ej. 10. Ejecutar esas mismas operaciones en un árbol 2-3.
Ej. 11. Demuestre por inducción las siguientes propiedades de árboles binarios y defina las funciones correspondientes en Haskell.

- Para todo árbol t: $alt.t \leq size.t$, en donde alt devuelve la altura y $size$ devuelve su tamaño (definir estas operaciones en Haskell).
- Para todo árbol t: $espejo.espejo.t = t$, en donde $espejo$ es la función que da vuelta los hijos de un árbol recursivamente (definirla en Haskell).
- Definir la función $mapTree : (a \to b) \to (Tree\ a) \to (Tree\ b)$, que dado un árbol, aplica una función dada a cada elemento del árbol.